Predicting time series - Neural ODE vs Reservoir computing

Albin Steen, Simone Piccioni, Viktor Olsson, Zachary Tio, Ole Fjeldså

Chalmers University of Technology

CHALMERS UNIVERSITY OF TECHNOLOGY

Problem formulation

- Today, diverse data varying in time and space is **crucial across fields like medicine, finance, and electronics**. Despite the significance of these fields, handling data in general as well as chaotic data poses challenges for prediction.
- One of the most prominent modern methods to predict data and especially chaotic data is **Reservoir computing**.
- This research explores how a rather new method in **Neural ODE** or **NODE** for short, compares to a reservoir computer in it's capability of predicting both more stable and chaotic data.
- This implementation aims to give insight into the performance of both NODE and a Reservoir computer as well as **how the two compare**. This is relevant and important as controlling and accurately predicting data is paramount for advancing work in mentioned critical domains.

Methods

GENERATING DATA: The data generation was done by iterating the dynamics for three systems; ***Van der pol, Lotka Volterra and Chaotic Lorenz attractor***. With these systems we get both chaotic time series and classical time series.

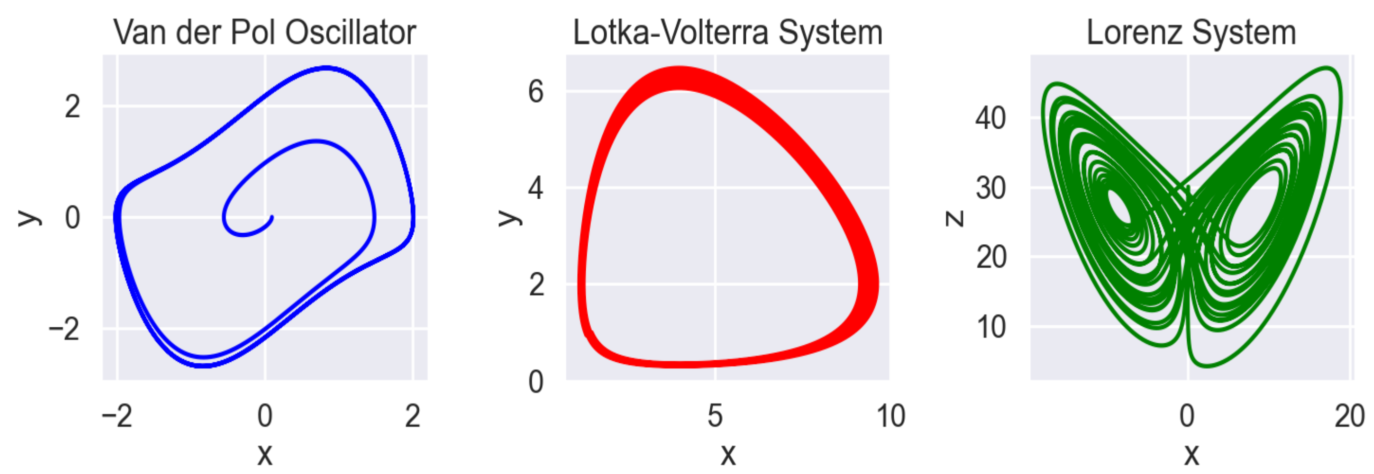

Figure 1: Examples of trajectories within the three systems used in the project

TIME PREDICTION: Reservoir Computing uses datasets to create activation states for training output connections. It depends on a fixed-weight Reservoir and a basic linear classifier Readout. This method offers inherent memory effects due to the Reservoir's recurrent connections. [3]

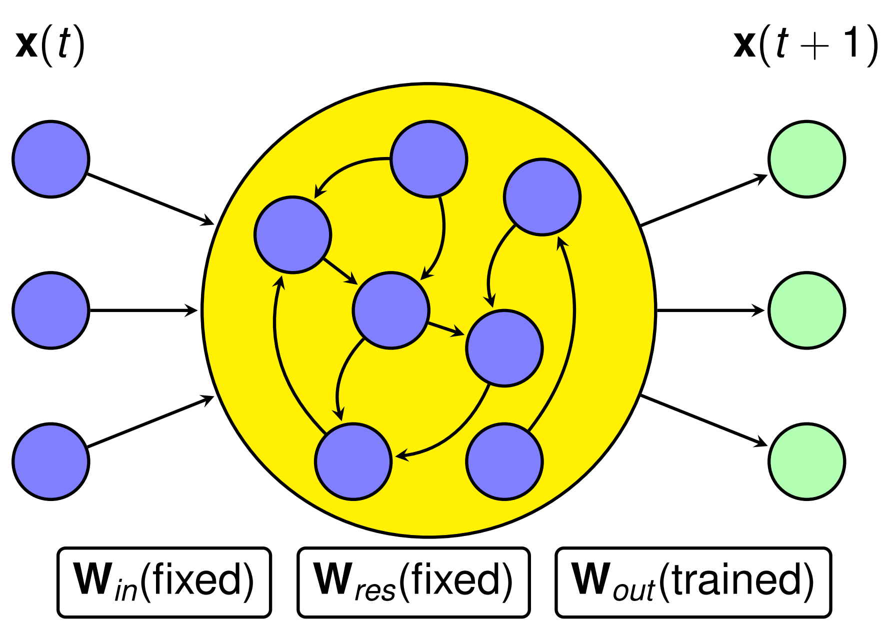

Figure 2: The typical structure of a reservoir computer

NODE is a relatively new class of deep neural network models that depart from the conventional discrete layer sequence. Instead of defining discrete hidden layers, they utilize a neural network to parameterize the derivative of the hidden state. The network's output is computed employing a black-box differential equation solver. To implement the NODE the library **"torchdiffeq"** and method **"neuralode"** was used.[2][1]

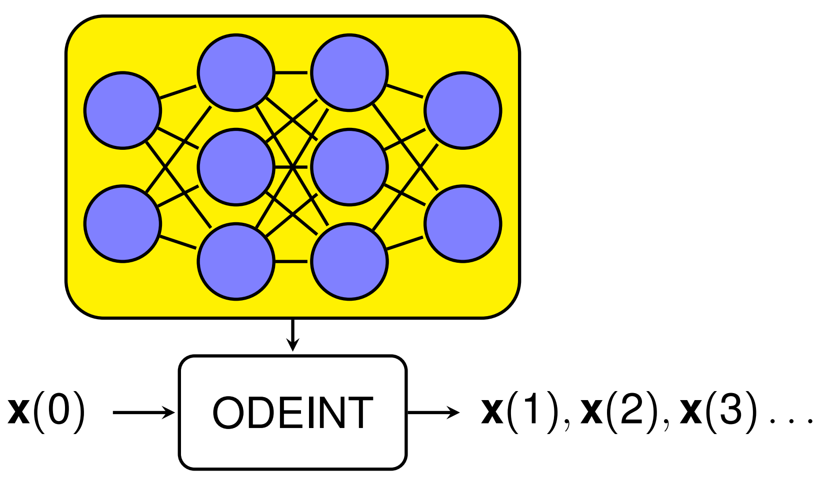

Figure 3: The typical structure for a neural ODE

Results

- **Van der Pol:**

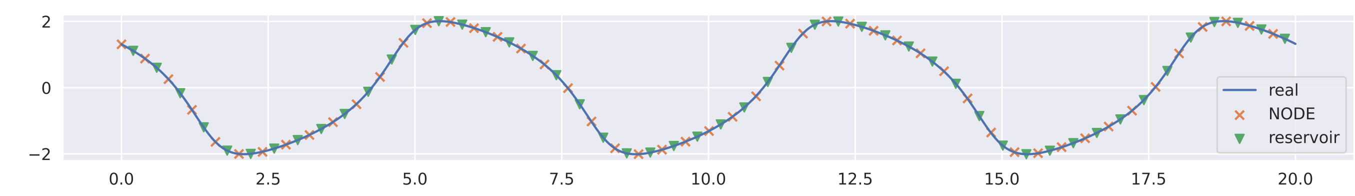

Figure 4: NODE and Reservoir on the first component of the Van der Pol system.

According to figure 4 we see that the NODE and reservoir are matched in performance for this type of time series.

- **Lotka Volterra:**

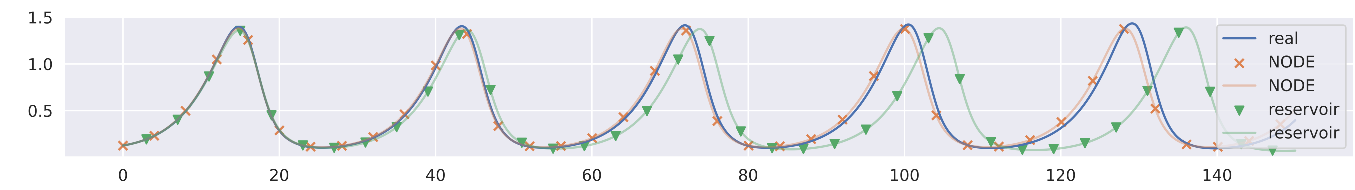

Figure 5: NODE and Reservoir on the first component of the Lotka-Volterra system. NODE predicted time:150*s*, Reservoir predicted time:73.7*s*.

For figure 5 we see that the NODE and reservoir computer predicts well but the NODE is more robust.

- **Chaotic Lorenz attractor:**

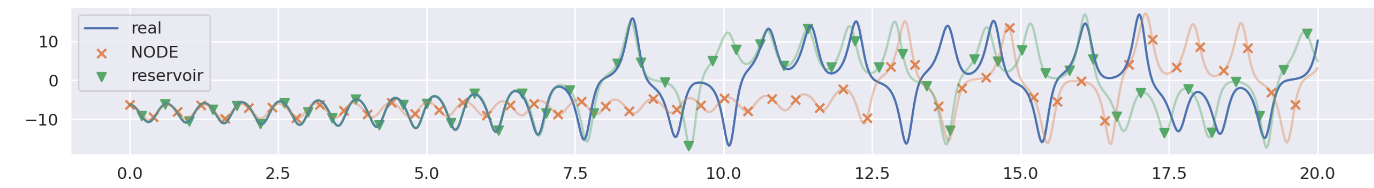

Figure 6: NODE and Reservoir on the first component of the Lorenz system. NODE predicted time:2.24*s*, Reservoir predicted time:7.5*s*.

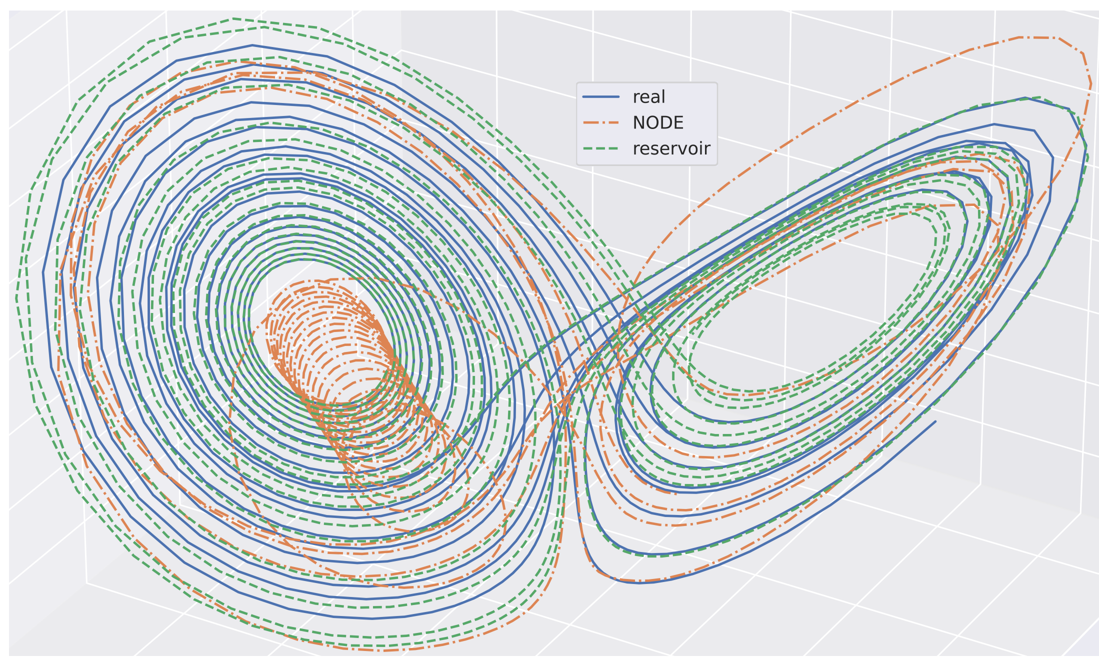

Figure 7: Both NODE and Reservoir are getting to the Lorenz attractor.

In contrast to the other results, according to figure 6 and 7, the reservoir outperforms the NODE for the more difficult and chaotic time series.

Conclusions - Future wor

- Both the reservoir computer and the NODE can predict less chaotic and more regular time series very good. For chaotic data, the NODE struggles a lot more compared to the Reservoir computer.
- The reservoir computer performs better with the chaotic data because it captures only the local dynamic of the system, while the NODE performs worse as it tries to get the global dynamic. Thus, reservoir computing is preferred for chaotic data.
- Unless the parameters for the reservoir computer are fine-tuned, the NODE has a slightly more robust behavior for the less chaotic data. Therefore, it could be argued that for this data NODE is preferred.
- **Future work:** A more methodic way to train NODEs could be explored, especially over chaotic datasets. Moreover other techniques, related to NODEs, can be analysed, such as *generative latent function time-series model* [2].

References

[1] R. T. Q. Chen.
torchdiffeq, 2018.

[2] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud.
Neural ordinary differential equations.
32nd Conference on Neural Information Processing Systems (NeurIPS 2018), 2018.

[3] L. Melandri.
Introduction to Reservoir Computing Methods.
PhD thesis, University of Bologna, 2014.